Generalisation de termes en theorie equationnelle. Cas associatif-commutatif

Loïc Pottier

UNITÉ DE RECHERCHE
INRIA-SOPHIA ANTIPOLIS

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P.105
78153 Le Chesnay Cedex
France
Tél. (1) 39 63 55 11

Rapports de Recherche

N° 1056

Programme 5

GENERALISATION DE TERMES EN THEORIE EQUATIONNELLE. CAS ASSOCIATIF-COMMUTATIF.

Loïc POTTIER

Juillet 1989

RR_1056

GENERALISATION DE TERMES EN THEORIE EQUATIONNELLE. CAS ASSOCIATIF-COMMUTATIF.

Loïc Pottier

INRIA Sophia Antipolis
2004 Route des Lucioles
06564 VALBONNE CEDEX

pottier@mirsa.inria.fr

Juin 1989

Résumé:

La généralisation de termes est la notion duale de l'unification. Contrairement à l'unification, elle a été très peu étudiée, essentiellement par [Reynolds 70], [Plotkin 70]et [Huet 76] , et uniquement sur les termes sans théorie équationnelle. La généralisation de termes trouve des applications dans la compilation de systèmes de règles de réécriture, dans l'induction et l'apprentissage automatique [Pottier 89].
On étudie ici la généralisation de termes dans une théorie équationnelle, d'abord dans le cas général, puis dans les deux cas de la théorie triviale et des théories Associatives-Commutatives, où on propose des systèmes de règles d'inférence rendant la généralisation effective (cf [Boudet + Contejean 88] pour un travail similaire sur l'unification AC) . On détaille ensuite le cas Associatif-Commutatif, pour y donner trois algorithmes de généralisation , dont deux s'appliquent à des cas particuliers, améliorant le système d'inférence de généralisation.

Abstract:

Terms generalization is the dual notion of terms unification. Few authors have studied it, principally [Reynolds 70], [Plotkin 70]et [Huet 76] , and only in the case of terms without equationnal theory. Terms generalization applies to compilation of terms rewriting systems, and to automatic induction and learning [Pottier 89].
We study here generalization of terms in equationnal theory, first in general case, and next in the two cases of trivial theory and of Associative-Commutative theories, where we propose two systems of inference rules which mecanize generalization (cf [Boudet + Contejean 88] for a similar work on AC unification). Finally we study in details the AC case, with three generalization algorithms (two applies to particular cases), which optimize the generalization system.

A. INTRODUCTION.

1. Notations, définitions.

T(F,V), l'algèbre des termes sur F un ensemble gradué de fonctions au plus dénombrable (la graduation est donnée par l'arité), et V un ensemble infini dénombrable de variables.
On utilisera les notations et définitions classiques [Huet 80], avec quelques précisions:
Une subtitution σ est notée $\{[X_1 \to t_1], \ldots, [X_n \to t_n]\}$.
Son support , qui est fini, est $Sup(\sigma) = \{X \in V \text{ tel que } \sigma X \neq X\}$ Elle est plate si les t_i sont des variables, et c'est un renommage si elle est bijective.
On dit que σ est un renommage sur un terme t si sa restriction à Var(t) est injective et à valeurs dans V.
La taille d'un terme est le nombre (avec répétitions éventuelles) de fonctions qui y apparaissent. On la note |t|. On a $|\sigma t| \geq |t|$, avec égalité si σ est plate.

2. Généralisations de termes dans une théorie équationnelle:

2.1. Définition: $(\overline{T}, \leq)$

Les substitutions définissent un préordre sur T par
$t \leq t' \Leftrightarrow \exists \sigma , \sigma t = t'$.
On dit alors que t est "plus général" que t'.
Ce préordre induit un ordre $\leq$ partiel sur $\overline{T} = T/R$ où R est la relation d'équivalence sur T définie par t R t' ssi $t \geq t'$ et $t \leq t'$. $\overline{T}$ est appelé l'ensemble des termes quotients. Il n'est autre que l'ensemble des classes d'équivalence de termes égaux à un renommage des variables près. Les éléments de $\overline{T}$ sont notés $\overline{t}$. Un renommage est plat donc on peut définir la taille d'un terme quotient , par $|\overline{t}| = |t|$.

2.2. Définition: $(renE(T), \leq_E)$:

Soit maintenant E une théorie équationnelle sur T , i.e. un ensemble d'équations entre termes. E définit une congruence sur T, notée $\simeq_E$. Les classes de $\simeq_E$ sont notées E(t), leur ensemble E(T) $(= T/\simeq_E)$. Les subtitutions donnent un préordre $\leq_E$ défini par

$$t \leq_E t' \Leftrightarrow \exists \sigma, \sigma t \simeq_E t'.$$

Ce préordre définit un ordre partiel $\leq_E$ sur T/R_E, où $t R_E t' \Leftrightarrow t \leq_E t'$ et $t \geq_E t'$.

On note renE(t) la classe de t modulo R_E, et renE(T) l'ensemble quotient T/R_E.

Remarquons que si E est vide , alors renE(T) = $\overline{T}$, et leurs ordres se confondent.

2.3.. Définition: généralisations principales.

On appelle généralisation principale de renE(t) et renE(t') tout minorant commun à ces deux termes, maximal parmi ces minorants communs.

L'ensemble des généralisations principales de renE(t) et renE(t') est noté renE(t)∧renE(t')

Par abus de langage, si renE(t) ∈ renE(t_1)∧renE(t_2), t sera appelé aussi généralisation principale de t_1 et t_2.

Le but de la généralisation est ici l'étude de renE(t)∧renE(t'). C'est le problème dual de l'unification dans une théorie équationnelle, qui consiste à déterminer les majorants minimaux de deux termes de renE(T).

B. CAS E = ∅: GENERALISATION DANS $\overline{T}$.

Si E est vide, alors renE(T) = $\overline{T}$, et leurs ordres se confondent.

On rappelle (et on en redémontre certains) quelques résultats de [Huet 76] et [Huet 80].

1.Proposition:

(i) Les minorants de $\overline{t}$ sont en nombre fini, et calculables ($\leqslant$ est donc bien fondé).
(ii) Deux éléments $\overline{t}$ et $\overline{t'}$ de $\overline{T}$ ont une unique généralisation principale, notée $\overline{t} \wedge \overline{t'}$.
(iii) Soit $\overline{c} = \overline{a} \wedge \overline{b}$, et u $\in$Occ(c) alors $\overline{c|_u} = \overline{a|_u} \wedge \overline{b|_u}$.
(iv) Deux termes de $\overline{T}$ ayant un majorant commun admettent une borne supérieure (qui correspond à un unificateur principal).

Preuve:

(i) par récurrence sur $|\overline{t}|$:

si $|\overline{t}|=0$, t$\in$V donc $\overline{t}$ n'a que lui-même comme minorant.

sinon $t = fa_i - fa_i$ dénote $f(a_1,\ldots,a_n)$ f étant d'arité n -

les minorants de $\overline{t}$ sont soit $\overline{X}$ -i.e. V - soit des termes de la forme $\overline{fb_i}$, avec $\forall i$, $b_i \leqslant a_i$

Lemme :

soient f d'arité n, $(\overline{t_i})$ n termes quotients,
alors $\{\overline{fb_i}$ tel que $\forall i\ \overline{b_i} = \overline{t_i}\}$ est fini.

Preuve du lemme:

On a $\overline{b_i}=\overline{t_i} \Leftrightarrow b_i = r_i\, t_i$, r_i renommage sur t_i. On peut choisir les t_i disjoints (i.e. sans variables communes) . On peut alors définir r telle que $\forall i\ rt_i=r_it_i$ et on a $\overline{fb_i} = \overline{rft_i}$. Seule la classe de rft_i importe donc on peut choisir r a valeurs dans un sous-ensemble fixe de V. De même on peut choisir les variables des t_i dans un sous-ensemble fini de V. Il n'y a alors
qu'un nombre fini de choix pour r, donc les $\overline{fb_i}$ sont en nombre fini, CQFD.

On a $|a_i|<|\overline{t}|$ donc, par hypothèse de récurrence, les minorants des $\overline{a_i}$ sont en nombre fini.

Or $\forall i\ \overline{b_i} \leqslant \overline{a_i}$ donc les n-uplets $(\overline{b_1},\ldots,\overline{b_n})$ sont en nombre fini, d'où , d'après le lemme1 , le nombre fini des $\overline{fb_i}$.

La calculabilité de ces minorants découle clairement de cette preuve.

(ii) et (iii):

On peut en trouver la preuve dans [Huet 76]. L'existence de la généralisation principale de deux termes, ainsi que sa contruction sont apparues dans [Reynolds 70] et [Plotkin 70].

Voici une définition constructive d'un terme $t = g(t_1, t_2)$ vérifiant $\overline{t} = \overline{t}_1 \wedge \overline{t}_2$ tirée de [Huet 80]:

On se donne une bijection φ entre T x T et V (tous deux infinis dénombrables):

- si $t_1 \in V$ ou $t_2 \in V$ ou t_1 et t_2 n'ont pas même racine, $g(t_1,t_2) = \varphi(t_1,t_2)$.
- si $t_1 = fa_i$, $t_2 = fb_i$, alors $t = fc_i$, où $\forall i, c_i = g(a_i,b_i)$.

La définition constructive de la généralisation principale donne un algorithme , en calculant φ par une liste d'instanciation des variables introduites incrémentale et globale. Cet algorithme est linéaire en la somme des tailles des termes de départ.

(iv)

Comme $\leqslant$ est bien fondé, l'ensemble des majorants de $\overline{t}$ et $\overline{t'}$, qui est non vide, a au moins un élément minimal (Zorn). Il est unique et minore tous les majorants de $\overline{t}$ et $\overline{t'}$ puisque deux termes ont toujours une borne inférieure (ii). C'est donc la borne supérieure de $\overline{t}$ et $\overline{t'}$.

Nous reprenons maintenant, et l'étendons, une idée proposée par J.P.Jouannaud [Jouannaud 88]. Elle consiste à voir la généralisation de termes comme l'utilisation de règles d'inférences, en s'inspirant de ce qui a été fait similairement pour le filtrage et l'unification, AC ou non [Boudet + Contejean 88].

Les deux problèmes de la généralisation en théorie triviale et en théorie AC seront envisagés sous cet angle.

2. Définition.

Un problème de généralisation G | S est constitué d'un terme G, et d'une suite d'équations S de la forme $u =_x v$, où x est une variable de G, qui représente une généralisation provisoire de u et v.

3. Le système d'inférence de généralisation SG :

Calculer la généralisation principale de deux termes u et v consiste à partir du problème $x \mid u =_x v$ et à appliquer les règles d'inférences suivantes (où x,y,z sont des variables, u,v, des termes, f une fonction n-aire):

RG_1: $\dfrac{G \mid f\,u_1 \ldots u_n =_x f\,v_1 \ldots v_n\,,\ S}{\{[x \to f\,z_1 \ldots z_n]\}\,G \mid u_1 =_{z1} v_1, \ldots, u_n =_{zn} v_n\,,\ S}$

où les z_i sont des nouvelles variables toutes distinctes.

RG_2: $\dfrac{G \mid u =_x v,\ u =_y v\,,\ S}{\{[x \to y]\}\,G \mid u =_y v,\ S}$

On notera $G \mid S \Rightarrow G' \mid S'$ le passage d'un problème de généralisation à un autre par l'une de ces règles.

4. Définition.

Une solution de $G \mid S$ est un triplet de substitutions (τ,σ,ρ) tel que pour toute équation $u =_x v$ de S, on a $\sigma\tau x = u$ et $\rho\tau x = v$.

τx est donc une généralisation, pas forcément principale, de u et v.

5. Lemme :

Soient $G \mid S \Rightarrow G' \mid S'$.

(i) $G < G'$.

(ii) Si (τ,σ,ρ) et (τ', σ', ρ') sont des solutions respectives de ces problèmes,

alors $\sigma\tau G = \sigma'\tau' G'$ et $\rho\tau G = \rho'\tau' G'$.

(iii) Pour toute (τ,σ,ρ) solution de $G \mid S$, il existe (τ', σ', ρ') solution de $G' \mid S'$, telle que $\tau G \leqslant \tau' G'$.

Preuve:

(i) est clair.

Deux cas se présentent, selon la règle utilisée.

Règle RG_1:

(ii) On a $G' = \{[x \to f z_1 \ldots z_n]\}\,G$. Par hypothèse, on a $\forall i$, $\sigma'\tau' z_i = u_i$, et $\sigma\tau x = f u_1 \ldots u_n$, donc il est clair que $\sigma\tau G = \sigma'\tau' G'$. La deuxième assertion est similaire.

(iii) on a $\sigma\tau x = f u_1 \ldots u_n$, et $\rho\tau x = f v_1 \ldots v_n$, donc

soit τx est une variable y, et alors $\tau' = \tau$, $\sigma' = \sigma \cup \{[z_1 \to u_1], \ldots, [z_n \to u_n]\}$,

$\rho' = \rho \cup \{[z_1 \to v_1], \ldots, [z_n \to v_n]\}$ conviennent,

soit $\tau x = f w_1 \ldots w_n$, et $\tau' = \tau \cup \{[z_1 \to w_1], \ldots, [z_n \to w_n]\}$, $\sigma' = \sigma$, $\rho' = \rho$ conviennent.

Règle RG_2:

(ii) est immédiate.

(iii) Il suffit de prendre $\tau' = \tau$, $\sigma' = \sigma$, $\rho' = \rho$.

On a alors le

6. Théorème:

Par le système SG, le problème $x \mid u =_x v$ a une unique forme normale $G \mid S$ (à un renommage de G près), et $\overline{G}$ est la généralisation principale de $\overline{u}$ et $\overline{v}$.

Preuve:

Par (i) et (ii) du lemme précédent, les réécritures de $x \mid u =_x v$ par SG donnent une suite strictement croissante de généralisations de u et v. Comme ces généralisations sont en nombre fini (cf B.1.(i)), cette suite est finie. Soit $G \mid S$ une forme normale de $x \mid u = x\ v$. Soit $\overline{Gen}$ la généralisation principale de $\overline{u}$ et $\overline{v}$, avec $\sigma Gen = u$, $\varrho Gen = v$. Considérons la solution $(\{[x \rightarrow Gen], \sigma, \varrho)$ de $x \mid u =_x v$. Par itération le long des applications des règles de SG du (iii) du lemme précédent, on obtient donc $Gen \leqslant \tau G$, où τ est un renommage (pour toute solution (τ, σ, ϱ) de $G \mid S$, τ est un renommage, car sinon on peut clairement appliquer une des deux règles R_1 ou R_2). Or $G \leqslant Gen$, donc $\overline{G} = \overline{Gen}$, et $\overline{G}$ est donc la généralisation principale de $\overline{u}$ et $\overline{v}$. CQFD.

7. Conclusion:

Calculer la généralisation principale de deux termes quotients $\overline{u}$ et $\overline{v}$ revient à calculer une forme normale quelconque du problème $x \mid u =_x v$ par le système SG.

8. Comparaison avec la méthode de G.Huet (cf 1. preuve):

La règle RG_2 fait en fait le travail de la bijection de $\overline{T} \times \overline{T}$ dans V de Huet, mais elle fait plus: elle permet de factoriser la généralisation de couples de sous-termes apparaissant plusieurs fois dans les deux termes de départ aux mêmes occurrences. La généralisation par SG sera donc plus rapide si les termes à généraliser ont des sous-termes répétés aux mêmes occurrences. Elle pourra en revanche être plus lente sinon, la bijection de Huet évitant parfois de conserver de nombreuses variables avec les mêmes instanciations, même si la méthode de Huet est un cas particulier d'application de SG avec un contrôle des règles favorisant la règle RG_2. Une implémentation de SG devrait permettre de trancher.

C. CAS GENERAL, THEORIES A CLASSES FINIES, THEORIES ALGEBRIQUES.

1. Proposition:

(i) $E(t) = E(t')$ ou $\overline{t} = \overline{t'} \Rightarrow renE(t) = renE(t')$

(ii) $E(t)$ est finie $\Rightarrow$ les minorants de $renE(t)$ sont en nombre fini,
$renE(t)$ contient $\overline{t}$, et est formée de la réunion d'un nombre fini de classes $\overline{t'}$.

Preuve:

(i) $E(t)=E(t') \Leftrightarrow t \simeq_E t' \Rightarrow \sigma t \simeq_E \sigma t' \simeq_E t' \simeq_E t$, σ étant la subtitution identité
$\Rightarrow renE(t)=renE(t')$.

$\overline{t} = \overline{t'} \Rightarrow \exists \rho$ renommage , $\rho t = t' \Rightarrow \rho t \simeq_E t' \Rightarrow renE(t)=renE(t')$.

(ii) $renE(t') \leqslant renE(t) \Leftrightarrow \exists \sigma, \sigma t' \simeq_E t \Leftrightarrow \exists \sigma, t'', \sigma t' = t'' \in E(t)$. Pour $t'' \in E(t)$ fixé les $\overline{t'}$ tels que $\exists \sigma, \sigma t' = t''$ sont en nombre fini (2. Proposition B.1.(i)). Or $\overline{t} = \overline{t'} \Rightarrow renE(t) = renE(t')$, donc les classes $renE(t')$ de tels t' sont aussi en nombre fini. Or $E(t)$ est finie , et le résultat est obtenu.

(i) prouve que $renE(t)$ contient $\overline{t}$.

$t' \in renE(t) \Rightarrow \exists \sigma, t''$, $\sigma t' = t'' \in E(t)$. Fixons t''. Les $\overline{t'}$ sont alors en nombre fini (Proposition B.1.(i)). Comme les t'' le sont aussi, les $\overline{t'}$ sont donc en nombre fini. Comme de plus $t' \in renE(t) \Rightarrow renE(t) = renE(t') \supset \overline{t'}$, $renE(t)$ est donc réunion d'un nombre fini de classes $\overline{t'}$.

2. Théories à classes finies.

Une théorie E est dite "à classe finies "si les classes d'équivalence $E(t)$ sont toutes finies.

2.1. Proposition:

Supposons E à classes finies . Alors $renE(t) \wedge renE(t')$ est fini et non vide.

Preuve: grâce à la proposition 1.(ii) il suffit de constater que $renE(X)$ est l'élément minimal de $renE(T)$.

Lorsque E est à classes finies énumérables, on sait tester l'ordre $\leqslant_E$, donc on dispose d'un algorithme trivial de détermination des généralisations principales, par énumération des minorants . Mais cet algorithme est de complexité rédhibitoire, par exemple lorque les classes de E sont de cardinal exponentiel en la taille de leurs éléments (cas des théories Associatives-Commutatives). En D. on propose trois algorithmes efficaces de calcul des

généralisations principales dans le cas Associatif-Commutatif, et un système d'inférence réalisant la généralisation de termes en théorie AC.

3. Théories algébriques.

E est constituée par des axiomes de commutativité et/ou d'associativité et/ou d'élément neutre et/ou d'idempotence de certaines fonctions de F. L'idempotence ou l'élément neutre rendent les classes infinies.

On a alors la difficulté suivante:

3.1. Théorème:

Soient $F = \{f,g,a,b\}$, a,b constantes, f et g binaires, et $E = \{(f(X,X)=X),(g(X,X)=X)\}$.
Alors il existe deux termes de renE(T) ayant une infinité de généralisations principales.

Preuve:

Soient $t=f(a,b)$, $t'=g(a,b)$, X et Y des variables de V,

$t_1=f(g(a,X),g(Y,b))$

$t_2=g(f(a,Y),f(X,b))$

soient s la substitution $\{[X\to a],[Y\to b]\}$ et $s'=\{[X\to b],[Y\to a]\}$

soit $g_0 = t_2$

et pour $n\geqslant 1$, $g_n = g(t_1, f(t_1, g_{n-1}))$

alors pour tout $n\geqslant 1$, $sg_n \simeq_E t$, et $s'g_n \simeq_E t'$ (donc $renE(g_n)$ est un minorant de renE(t) et renE(t')).

Si $n\neq m$, il est clair que $renE(g_n)$ et $renE(g_m)$ sont incomparables par $\leqslant$ et $renE(g_n)$ est maximal parmi les minorants de renE(t) et renE(t') (car si on substitue X par autre chose que a dans g_n, les simplifications conduisant à t ne peuvent avoir lieu).

En résumé les $renE(g_n)$, $n\geqslant 1$ forment une infinité de généralisations principales de renE(t) et renE(t').

Une conséquence de ceci est qu'il peut y avoir une infinité de généralisations principales de deux formules booléennes (si on écarte la distributivité des connecteurs logiques).

D. GENERALISATION ASSOCIATIVE-COMMUTATIVE.

On propose un système de règles d'inférence (du type du système SG de B.) rendant effective la généralisation AC.
Puis , après une étude des termes AC et du filtrage de ces termes, on propose trois algorithmes de généralisation complétant ce système d'inférence:

- le premier est l'algorithme général déterminant les généralisations principales de deux termes, ou d'un nombre fini de termes.

- le second est plus rapide , mais se limite à chercher les généralisations maximales parmi les généralisations linéaires de deux termes (resp. d'un nombre fini de termes).

- le troisième traite le cas des termes de la forme $f(x_1,\ldots,x_n)$, où f est associative commutative, et les x_i sont des variables. Il provient d'une étude qu'on mène des semi-groupes de $(IN^*, +)$, où on propose quelques résultats sur le test d'appartenance à de tels semi-groupes.

Les algorithmes proposés ont été écrits en Le_Lisp sur Macintosh SE et sur le Gould de l'INRIA Sophia Antipolis. Le système d'inférence n'a pas encore été implémenté. En annexe on donne des exemples d'éxécutions (les temps sont ceux de Le_Lisp compilé par le compilateur Complice sur le Gould de l'INRIA Sophia Antipolis) montrant l'efficacité et les limites des trois algorithmes décrits.

1. Préliminaires:

On étudie ici les termes Associatifs-Commutatifs, leur filtrage, et on donne des propriétés limitatives sur leurs généralisations. Ce qui concerne le filtrage AC et les limitations sur la généralisation AC peut être sauté en première lecture.

La théorie équationnelle E qu'on considère est donc constituée des axiomes de commutativité et d'associativité de certaines fonctions binaires de F (dites "fonctions AC", dont l'ensemble est noté F_{ac}). Les classes E(t) sont alors finies, bien qu'ayant de nombreux éléments. Par exemple, si f est AC, la classe $E(f(a_1, f(a_2, ..., f(a_{n-1}, a_n)...))$ a $\frac{(2n-2)!}{(n-1)!}$ éléments. Les résultats de C.2. sur les théories à classes finies s'appliquent donc ici, en particulier deux termes de renE(T) ont un nombre fini non nul de généralisations principales. On a quelques propriétés spécifiques:

1.1. Propriété:

a. renE(t) est la réunion des $\overline{t'}$ où t' parcourt E(t).

b. renE(t) est la réunion des E(t') où t' parcourt $\overline{t}$.

c. $t' \in renE(t) \Leftrightarrow \exists \rho$ renommage sur t' tel que $\rho t' \in E(t)$.

d. les termes de renE(t) ont tous la même taille.

1.2. Notations.

La congruence engendrée par E est ici notée $\simeq_{ac}$: $t \simeq_{ac} t' \Leftrightarrow E(t) = E(t')$.

Le préordre $\leqslant_E$ est noté $\leqslant_{ac}$: $t \leqslant_{ac} t' \Leftrightarrow \exists \sigma, \sigma t \simeq_{ac} t'$.

Une substitution est représentée par un ensemble de substitutions élémentaires ($[x \rightarrow a]$ où x est une variable et a est un terme AC).

Deux termes t et t' tels que renE(t) = renE(t') sont dits "équivalents modulo AC et renommage":

$renE(t) = renE(t') \Leftrightarrow t \leqslant_{ac} t'$ et $t' \leqslant_{ac} t \Leftrightarrow \exists \rho$ renommage, $\rho t \simeq_{ac} t'$.

Une classe E(t) est appelée "terme AC".

En considérant les fonctions AC comme des fonctions à nombre variable d'arguments (supérieur à 2), sans ordre sur les arguments, on représente généralement un terme AC par une forme canonique [Hullot 80], [Le Chenadec 83]qui permet de décider $\simeq_{ac}$. Cela consiste à "applatir" un terme de racine AC (ses arguments n'ont alors plus la même racine que le terme: ainsi f(a,f(b,c)) donne f(a,b,c)), à ordonner ses arguments mis eux aussi sous forme canonique (par un ordre total sur les collections - ou multi-ensembles -), puis à regrouper les

arguments égaux. Par exemple , si f est AC et g non AC, et si t = f(g(a,b),f(f(b,a),b)), E(t) est représenté par la forme canonique de t: $f(a^1,b^2,g(a,b)^1)$.
On notera alors un terme AC de racine f AC par $f\,a1^{u1} \ldots an^{un}$, ou par $f\,a^u$, les a_i sont des termes AC différents deux à deux, les u_i des entiers naturels non nuls , a le vecteur $(a_1,\ldots,a_n)$ et u le vecteur $(u_1,\ldots,u_n)$. Par extension, le terme $f\,a1^{u1} \ldots an^{un}$,où tous les u_i sont nuls sauf u_k qui est égal à 1 , dénote alors le terme a_k.

1.3. Filtrage AC.

Le problème du filtrage AC est de trouver , étant donnés deux termes t et t', une ou les substitutions σ telles que $\sigma t \simeq_{ac} t'$ (une suffit si on veut simplement tester $t \leqslant_{ac} t'$).
Différents algorithmes existent, qui ont commun d'utiliser les formes canoniques AC et la résolution d'équations diophantiennes linéaires (par exemple [Hullot 80], [Le Chenadec 83]. On en propose un similaire, avec une amélioration nouvelle qui utilise une condition nécessaire au filtrage portant sur l'appartenance à un semi-groupe de (IN*,+), notion développée par la suite pour la généralisation AC au §5.

a. Définition:

Les substitutions $\sigma_1,\ldots,\sigma_n$ sont "ac-compatibles " ssi $\forall X \in V$, $\sigma_1(X) \simeq_{ac} \ldots \simeq_{ac} \sigma_p(X)$.

b. Propositions:

Soient E(t) = fa^u et E(t') = fb^v deux termes AC, où f est AC, $u = (u_1,\ldots,u_p)$, et $v = (v_1,\ldots,v_q)$:

b1.

$t \leqslant_{ac} t'$
$\Leftrightarrow$ il existe $L = (\lambda_{ij})$ une matrice de $M_{q,p}(IN)$, sans colonne nulle, et il existe $\sigma_1,\ldots,\sigma_p$ des substitutions ac-compatibles, vérifiant:
- $Lu = v$
- $\forall i,\ \sigma_i a_i \simeq_{ac} f\,b1^{\lambda 1 i} \ldots bq^{\lambda q i}$

Preuve: dans [Hullot 80]

On peut ici remarquer que si a_i n'est pas une variable, alors on doit avoir $\sum_j \lambda_{ji} = 1$, et ainsi si $\lambda_{ki} = 1$, b_k et a_i ont même racine. Ce qui permet de limiter le nombre des matrices L candidates, parmi celles qui n'ont pas de colonne nulle, et qui vérifient Lu = v.

b2.Corollaire:

si t est linéaire (i.e. chaque variable de t n'a qu'une seule occurrence) :
$t \leqslant_{ac} t'$
$\Leftrightarrow$ il existe $L=(\lambda_{ij})$ une matrice de $M_{q,p}(IN)$, sans colonne nulle vérifiant:
- $Lu=v$
- $\forall i, a_i \leqslant_{ac} f\, b_1^{\lambda 1 i} \ldots b_q^{\lambda q i}$

Dans ce cas , l'ac-compatibilité des substitutions de b1. est triviale.

c. Condition nécessaire:

Avec les notations précédentes, il est immédiat que:

$t \leqslant_{ac} t' \Rightarrow \forall j, v_j$ appartient au semi-groupe de $(IN^*,+)$ engendré par les $u_1,\ldots,u_p$.

L'utilisation de cette condition nécessaire constitue l'originalité de notre algorithme, et son test efficace est décrit au §5.

d. Algorithme:

On utilise 4 fonctions, qui ont les caractéristiques suivantes:
- MATRICES(a,u,b,v) : donne toutes les matrices $L = (\lambda_{ij})$ sans colonne nulle vérifiant $Lu=v$, et si $a_i \notin V$, il existe un unique k tel que $\lambda_{ki} = 1$, et b_k et a_i ont même racine.
- FILTRE(t,t') : donne toutes les subtitutions σ telles que $\sigma t \simeq_{ac} t'$.
- FILTREL($(t_1, t'_1), \ldots, (t_n, t'_n)$) : donne toutes les substitutions σ vérifiant pour tout i, $\sigma t_i \simeq_{ac} t'_i$.
- TEST (u,v) : teste quelques conditions nécessaires à l'existence d'une matrice L sans colonne nulle telle que $Lu = v$ (qui sont données au §5, et qui comprennent la condition nécessaire c. précédente)

La fonction MATRICE est réalisée par un algorithme proche de celui qu'on exposera au §5 (décision de l'ordre $\leqslant$ sur les partitions d'entiers).
Les algorithmes de FILTRE et FILTREL sont les suivants:

FILTRE(t,t'):
si $t \in V$, rendre $\{\{[t \to t']\}\}$
si t et t' n'ont pas même racine , rendre $\varnothing$
si $t = g\,a_1 \ldots a_n$, $t' = g\,b_1 \ldots b_n$, où g est non AC,
rendre FILTREL$((a_1, b_1), \ldots, (a_n, b_n))$
si $t = f a^u$, $t' = f b^v$, où f est AC: $u = (u_1, \ldots, u_p)$, $v = (v_1, \ldots, v_q)$
si TEST(u,v) = faux, rendre $\varnothing$
sinon soient LM = MATRICES(a,u,b,v)
et LSUB = $\varnothing$ dans
pour chaque matrice $L = (\lambda_{ij})$ de LM:
soit $\forall i$, $c_i = f\, b1^{\lambda 1 i} \ldots bq^{\lambda q i}$ dans
LSUB := LSUB $\cup$ FILTREL $((a_1, c_1), \ldots, (a_p, c_p))$
rendre LSUB.

FILTREL$((t_1,t'_1), \ldots, (t_n, t'_n))$:
soit LSUB1 := FILTRE (t_1, t'_1)
si LSUB1 $\neq \varnothing$, rendre $\varnothing$
sinon si n = 1, rendre LSUB1
sinon soit LSUB2 = FILTREL$((t_2,t'_2), \ldots, (t_n, t'_n))$ dans
rendre $\bigcup_{\sigma \in \text{LSUB1}} \{ \sigma \cup \sigma'$, où $\sigma' \in$ LSUB2 et σ' ac-compatible avec $\sigma \}$.

Remarque: nous avons choisi dans FILTREL de ne pas propager les substitutions de FILTRE (t_1, t'_1) sur $(t_2,t'_2), \ldots, (t_n, t'_n)$, pour éviter des calculs de mise sous forme canonique AC des termes substitués, qui peuvent être coûteux.

1.4. Généralisation AC, limitations.

Rappelons que deux termes renE(t) et renE(t') de renE(T) ont un nombre fini non nul de généralisations principales, dont l'ensemble est noté renE(t) $\wedge$ renE(t'). Pour les calculer, le premier algorithme qui vient à l'esprit est de déterminer les minorants de renE(t) (qui sont en nombre fini), d'éliminer ceux qui ne minorent pas renE(t') (par tests de filtrage AC), puis de ne garder que ceux qui ne sont inférieurs à aucun autre (toujours par filtrage AC). Cet algorithme est impraticable, les minorants d'un terme AC même simple étant trop nombreux : par exemple le terme renE(f(X, f(X,...,f(X, X)...)) , où X apparaît n fois, a $P(1) + \ldots + P(n)$ minorants dans renE(T), où P(m) est le nombre de partitions de l'entier m, i.e. le nombre de solutions de l'équation $x_1 + 2x_2 + \ldots + mx_m = m$.

On est alors conduit à effectuer de trop nombreux tests de filtrage AC, opération la plus coûteuse, avec l'unification AC, dans la manipulation de termes AC.

La propriété suivante donne la trame d'un autre algorithme:

a. Propriété:

$renE(t_1) \wedge renE(t_2)$ est constitué des éléments maximaux de l'ensemble

$\{ renE(t'') \mid \overline{t''} = \overline{t'}_1 \wedge \overline{t'}_2, \forall i=1,2, t'_i \simeq_{ac} t_i \}$.

Preuve: nous aurons besoin d'un lemme:

Lemme:

$\forall t,t', \sigma t=t' : renE(t) = renE(t') \Leftrightarrow \sigma$ est un renommage sur t.

Preuve du lemme:

La condition suffisante résulte de la propriété 1.1.c.

Condition nécessaire: tous les termes de renE(t) ont même taille et même nombre de variables. Donc si σ n'est pas un renommage sur t, t' est de taille strictement inférieure à celle de t, ou bien a moins de variables, et donc $renE(t) \neq renE(t')$.

Revenons à la propriété:

Il suffit de montrer $renE(t) \in renE(t_1) \wedge renE(t_2) \Rightarrow \overline{t} = \overline{t'}_1 \wedge \overline{t'}_2$, où $t'_1 \in E(t_1)$ et $t'_2 \in E(t_2)$.

Il est clair que $renE(t) \in renE(t_1) \wedge renE(t_2) \Rightarrow \exists t'_1, t'_2, \overline{t} \leqslant \overline{t'}_1 \wedge \overline{t'}_2, \forall i\ t'_i \in E(t_i)$.

Supposons que $\sigma t = t'$, σ n'étant pas un renommage sur t (i.e. $\overline{t} < \overline{t'}$), avec $\overline{t'} \leqslant \overline{t'}_1 \wedge \overline{t'}_2$.

Alors $\forall i$, $renE(t') \leqslant renE(t'_i) = renE(t_i)$, et $renE(t) < renE(t')$ (par le Lemme et car $renE(t) \leqslant renE(t')$),

donc $renE(t) \notin renE(t_1) \wedge renE(t_2)$. Et ainsi $t = \overline{t'}_1 \wedge \overline{t'}_2$.

L'algorithme qui en découle est le suivant: on calcule les classes $E(t_1)$ et $E(t_2)$, puis l'ensemble $\{ renE(t'') \mid \overline{t''} = \overline{t'}_1 \wedge \overline{t'}_2, \forall i=1,2, t'_i \simeq_{ac} t_i \}$, dont on calcule les éléments maximaux par filtrage AC. Cet algorithme est meilleur, mais est encore inutilisable, du fait de la taille des classes E(t).

Avant de proposer des algorithmes de généralisation AC plus efficaces et utilisables, nous donnons quelques remarques et propriétés sur la complexité de la généralisation AC et sur l'existence éventuelle d'algorithmes qui soient récursifs sur la structure des termes, comme c'est le cas avec la théorie triviale .

Pour alléger les notations, dans la suite on appellera "généralisation AC" aussi bien un élément renE(t) de $renE(t_1) \wedge renE(t_2)$, que le terme t ou le terme AC E(t). On cherchera alors des représentants dans T ou dans E(T) des généralisations principales de deux termes.

b. Remarque:

Les généralisations AC n'ont pas forcément la même taille (nombre de symboles fonctionnels)
Exemple: si f est AC, les termes AC f aab et f ab'b' ont deux généralisations AC,
qui sont f aXY et f XXY, de tailles 2 et 1 respectivement.

c. Proposition (nombre de généralisations AC):

Le nombre de généralisations AC est exponentiel en la taille des termes de départ.

Preuve:

soient f une fonction AC, t_1 = f aab, t_2 = f acc, et h une fonction n-aire
soit g_0 = f aZY et g_1 = f XXY les deux généralisations AC de t_1 et t_2.
soit $t_k = h(g_{i1}, \dots, g_{in})$, $0 \leqslant k \leqslant 2^n-1$, $(i_1,\dots,i_n)$ étant l'écriture binaire de k.
Alors les t_k sont les généralisations AC de $t'_1 = h(t_1,\dots,t_1)$ et $t'_2 = h(t_2,\dots,t_2)$, et sont au nombre de 2^n.
On a donc, pour tout $n \geqslant 1$, deux termes de taille $4n+1$ qui ont 2^n généralisations AC.

La propriété 1.4.a. permet de majorer le nombre de généralisations AC de t et t' par #E(t).
#E(t') (on note #E le cardinal d'un ensemble E). On montre facilement que
$\#E(t) \leqslant \frac{(2\|t\| - 2)!}{(\|t\| - 1)!}$, où $\|t\|$ est la taille de t en comptant aussi les variables. Ainsi

$$\#\text{renE}(t) \wedge \text{renE}(t') \leqslant \left(\frac{(2n-2)!}{(n-1)!} \right)^2 , \text{ où } n = \sup \{\|t\|, \|t'\|\}. \text{ CQFD}$$

On va maintenant présenter trois algorithmes de généralisation AC utilisables en pratique, l'un pour les termes décrits en d1., les autres généraux.

d. Etude du lien généralisations d'un terme <-> généralisations des sous-termes: théorèmes de structures.

d1. Premier cas: généralisations de deux termes de même racine non AC.

Soient t_1 = f $a_1 \dots a_n$ et t_2 = f $b_1 \dots b_n$ avec f non AC. Il est clair qu'un minorant non variable de t_1 et t_2 a la forme f $c_1 \dots c_n$.

Théorème 1:
$\text{renE}(fc_i) \in \text{renE}(t_1) \wedge \text{renE}(t_2)$ n'entraîne pas $\forall i$, $\text{renE}(c_i) \in \text{renE}(a_i) \wedge \text{renE}(b_i)$
(contrairement à la théorie triviale E=∅),
mais implique que $\forall i$, $\exists a'_i \in E(a_i)$, $\exists b'_i \in E(b_i)$, tels que $\overline{c_i} = \overline{a'_i} \wedge \overline{b'_i}$.

Preuve:

- un contre-exemple montre la première assertion:
t_1 = f g(a a b) g(a a c), t_2 = f g(a' a' b') g(a' b' b') , avec g AC.

alors les généralisations principales sont

t_3= f g(X X Y) g(X Z T)

t_4=f g(X X Y) g(Z Z T)

t_5= f g(X Y Z) g(X Y T)

t_6=f g(X Y Z) g(Y Y T)

on a renE(g(a a b))$\wedge$renE(g(a' a' b'))=renE(g(a a c))$\wedge$renE(g(a' b' b'))={renE(g(X X Y))}

mais si $fc_i = t_3$, renE(c2)=renE(g(X Z T))$\notin$renE(g(a a c))$\wedge$renE(g(a' b' b')).

- seconde assertion: d'après la propriété 1.4.a. on a $\overline{fc_i}=\overline{fa'_i}\wedge\overline{fb'_i}$, avec les conditions données sur les a'_i et b'_i.

On a donc $\forall i, \overline{c_i}=\overline{a'_i}\wedge\overline{b'_i}$ (cf généralisation en théorie triviale)

CQFD.

On n'a donc pas de calcul directement récursif sur la structure des termes possible de l'ensemble des généralisations principales. Par contre:

Théorème 2:

Soient $c_1,\ldots,c_n$ des termes sans variables communes et f non AC.

Si $\forall i$, renE(c_i)$\in$renE(a_i)$\wedge$renE(b_i), alors il existe $d_1,\ldots,d_n$ tels que $\forall i$ renE(d_i)=renE(c_i) et renE(fd_i) $\in$ renE(fa_i)$\wedge$renE(fb_i).

Preuve:

On a $\forall i, \exists\sigma_i,\tau_i$, $\sigma_i c_i \in E(a_i)$, et $\tau_i c_i \in E(b_i)$. Les c_i sont disjoints donc on peut prolonger les σ_i et τ_i en σ et τ telles que $\forall i$ $\sigma c_i \in E(a_i)$, et $\tau c_i \in E(b_i)$. Or $f \notin F_{ac}$, donc $\sigma fc_i \in E(fa_i)$ et $\tau fc_i \in E(fb_i)$, d'où renE(fc_i) est un minorant de renE(fa_i) et renE(fb_i). Donc il existe fd_i tel que renE(fc_i)$\leqslant$renE(fd_i)$\in$ renE(fa_i)$\wedge$renE(fb_i). Mais alors, comme $f \notin F_{ac}$, $\forall i$, renE(c_i)$\leqslant$renE(d_i)$\leqslant$renE(a_i) et renE(c_i)$\leqslant$renE(d_i)$\leqslant$ renE(b_i).

Donc, comme $\forall i$ renE(c_i)$\in$renE(a_i)$\wedge$renE(b_i), pour tout i, renE(c_i)=renE(d_i).

CQFD.

On peut donc déterminer une généralisation principale de renE(t) et renE(t') à partir de généralisations principales de leurs premiers sous-termes.

d2. Deuxième cas : généralisation de deux termes de même racine AC.

Il est clair qu'une généralisation AC de ces termes a la forme fc^w.

Notons $\leqslant$ la relation définie sur les vecteurs d'entiers par $u \leqslant v \Leftrightarrow \exists$ L une matrice d'entiers naturels, sans colonne nulle, telle que Lu = v. On verra au §5 que, modulo l'ordre des coefficients de u et v, $\leqslant$ est une relation d'ordre, et que deux vecteurs ont un nombre fini non nul de minorants communs maximaux, qu'on note $u\wedge v$. On a clairement $t_1 \leqslant_{ac} t_2 \Rightarrow u \leqslant v$ (cf §1.3.b.).

Théorème 3:

On suppose que a , b et c sont des vecteurs de variables distinctes. Alors:

(i) $t_1 \leqslant_{ac} t_2 \Leftrightarrow u \leqslant v$

(ii) $renE(fc^W) \in renE(fa^U) \wedge renE(fb^V) \Leftrightarrow w \in u \wedge v$.

Preuve: elle résulte du lemme:

Lemme: soit a et b des vecteurs de variables distinctes,

alors $renE(fa^u) \leqslant renE(fb^v) \Leftrightarrow u \leqslant v$.

Preuve:

La condition nécessaire vient de la propriété 1.3.b1. Inversement cette propriété donne la définition des substitutions (alors ac-compatibles car de supports disjoints) , qui réalisent $renE(fa^u) \leqslant renE(fb^v)$.

On étudiera en détail au §5. la généralisation AC de telles termes, qui se ramène donc au calcul de $u \wedge v$.

Ce théorème ne se généralise pas:

Théorème 4:

$renE(fc^W) \in renE(fa^U) \wedge renE(fb^V)$ n'implique pas $w \in u \wedge v$.

Preuve:

un contre-exemple:

t_1 = f baa et t_2 = f acc ont deux généralisations principales

qui sont f aXY et f XXY. Dans le cas de la première, on a w = (1,1,1) et u=v=(1,2), donc $u \wedge v$ = {u}, et $w \notin u \wedge v$.

CQFD.

En conclusion, ces théorèmes interdisent l'existence d'algorithmes de généralisation AC directement récursifs sur la structure des termes.

2. Le système d'inférence SGAC de généralisation AC.

Dans ce qui suit tous les termes seront des termes AC.

De manière similaire au cas de la théorie triviale, on définit un système de règles d'inférence , appelé SGAC constitué des règles suivantes (où f, g sont des fonctions, x,y,z des variables, G,a,b,u,v des termes AC, A,A',B,B',C des collections de termes AC non vides):

$$RGAC_1 : \quad \frac{G \mid fAA' =_x fBB', S}{\{[x \to fyz]\}G \mid fA =_y fB, fA' =_z fB', S} \qquad \text{où f est AC, y et z de nouvelles variables.}$$

$$RGAC_2: \quad \frac{G \mid ga_1 \ldots a_n =_x gb_1 \ldots b_n, S}{\{[x \to gy_1 \ldots y_n]\}G \mid a_1 =_{y1} b_1, \ldots, a_n =_{yn} b_n, S}$$

où g est n-aire non AC, $y_1, \ldots, y_n$ de nouvelles variables.

$$RGAC_3: \quad \frac{G \mid u =_x v, u =_y v, S}{\{[x \to y]\}G \mid u =_y v, S}$$

La notion de solution est la même que pour la théorie triviale (B.).

On a alors les deux lemmes suivants:

2.1. Lemme:

(i) Si $G \mid S \Rightarrow G' \mid S'$, alors $G <_{ac} G'$, et si (τ, σ, ϱ) et $(\tau', \sigma', \varrho')$ sont des solutions respectives de ces problèmes, $\sigma\tau G = \sigma'\tau'G'$ et $\varrho\tau G = \varrho'\tau'G'$.

(ii) Soit $x \mid u =_x v \Rightarrow^* G \mid S$. Pour toute solution (τ, σ, ϱ) de $G \mid S$ où τ n'est pas un renommage sur G, il existe $G' \mid S'$ et une solution $(\tau', \sigma', \varrho')$ de $G' \mid S'$, tels que $G \mid S \Rightarrow G' \mid S'$ et $\tau G = \tau' G'$.

Preuve:

(i) $G <_{ac} G'$ est clair pour chaque règle. Prouvons le reste de l'assertion pour chaque règle.

Règle $RGAC_1$: on a $\sigma'\tau'G' = \sigma'\tau'\{[x \to fyz]\}G = \sigma\tau\{[x \to fAA']\}G = \sigma\tau G$, de même pour la seconde assertion.

Règle $RGAC_2$: on a $\sigma'\tau'G' = \sigma'\tau'\{[x \to gy_1 \ldots y_n]\}G = \sigma\tau\{[x \to ga_1 \ldots a_n]\}G = \sigma\tau G$, de même pour la seconde assertion.

Règle $RGAC_3$: on a $\sigma'\tau'G' = \sigma'\tau'\{[x \to y]\}G = \sigma\tau\{[x \to u]\}G = \sigma\tau G$, de même pour la seconde assertion.

(ii) τ n'est pas un renommage, donc un au moins des trois cas suivants se présente:

1. Il existe deux variables de G, x et y avec $\tau x = \tau y = z$:
alors $RGAC_3$ s'applique et $\tau' = \tau$, $\sigma' = \sigma$, $\varrho' = \varrho$ conviennent.

2. Il existe une variable x de G avec $\tau x = gc_1 \ldots c_n$:

alors $RGAC_2$ s'applique et $\tau' = \tau \cup \{[y_1 \rightarrow c_1],\ldots,[y_n \rightarrow c_n]\}$, $\sigma' = \sigma$, $\varrho' = \varrho$ conviennent.

3. Il existe une variable x de G avec $\tau x = fC$:

alors $RGAC_1$ s'applique et $\tau' = \tau \cup \{[y \rightarrow c],[z \rightarrow fC]\}$, $\sigma' = \sigma$, $\varrho' = \varrho$ conviennent.

2.2. Lemme:

Soit $x \mid u =_x v \Rightarrow^* G \mid S$.

(i) G est une généralisation AC de u et v.

(ii) il n'existe pas de suite infinie $x \mid u =_x v \Rightarrow \ldots \Rightarrow G \mid S \Rightarrow \ldots$

Preuve:

(i) Le (i) du lemme précédent appliqué à chaque réécriture par SGAC prouve que G est une généralisation AC de u et v.

(ii) Le (i) du lemme précédent montre que les généralisations obtenues lors de ces réécritures forment une suite strictement croissante de généralisations AC de u et v. Celles-ci étant en nombre fini (cf D.1.), la suite de réécriture ne peut être infinie.

On a alors le

2.3. Théorème:

Toutes les généralisations AC principales G de u et v sont (modulo renommage) des formes normales de $x \mid u =_x v$.

Preuve:

Soit Gen une généralisation AC principale de u et v.

Il existe alors une solution $(\{[x \rightarrow Gen]\}, \sigma_0, \varrho_0)$ au problème $x \mid u =_x v$. On itère alors le (ii) du lemme 2.1. jusqu'à obtenir $x \mid u =_x v \Rightarrow^* G \mid S$ avec une solution (τ, σ, ϱ) où τ est un renommage. On a alors $Gen = \tau G$. G est de plus une forme normale, car sinon ((i) du lemme 2.2.) il existerait une généralisation AC de u et v strictement supérieure.

2.4. Conclusion:

Il est clair que chaque règle d'inférence n'a qu'un nombre fini de façons de s'appliquer à un problème. Avec le (ii) du lemme 2.2. on en déduit que les formes normales par SGAC d'un problème $x \mid u = x\ v$ sont en nombre fini. Les généralisations AC principales de u et v sont alors les éléments maximaux pour $\leqslant_{ac}$ des formes normales de $x \mid u =_x v$ par SGAC.

2.5. Utilisation de différentes stratégies d'application des règles du système SGAC:

Dans les trois paragraphes qui vont suivre, on va décrire trois algorithmes de généralisation AC, le premier étant spécifique à une certaine classe de termes AC simples, les deux derniers correspondant à une optimisation du système d'inférence SGAC utilisé avec deux stratégies particulières d'application des règles d'inférences.

3. GENERALISATION AC. CAS GENERAL.

On propose ici un algorithme de calcul des généralisations AC de deux termes AC quelconques, dont on montrera qu'il peut être vu comme une utilisation du système SGAC avec une stratégie particulière d'application de ses règles d'inférence, et où la règle $RGAC_1$ est quelque peu modifiée.

On définit d'abord quelques notions essentielles pour la suite.

3.1. Collections, b-correspondances et a-correspondances.

a.Def:

une collection d'un ensemble E est une application c de E dans IN.
Son cardinal est $|c| = \sum_{x \in E} c(x)$, et elle est dite vide si $|c| = 0$. Si $c(x) \neq 0$ on notera $x \in c$. On définit naturellement l'inclusion, la différence, l'intersection, et l'inclusion de collections de E par:

- $c \subset c' \iff \forall x \in E, c(x) \leqslant c'(x)$ (c est alors une "sous-collection de c").
- $\forall x \in E$,
  $(c-c')(x) = \sup \{0, c(x) - c'(x)\}$
  $(c \wedge c')(x) = \inf\{c(x), c'(x)\}$
  $(c \cup c')(x) = c(x) + c'(x)$.

Dans la suite E sera fini, et on notera plutôt une collection c de E par $c = (a_i)_{1 \leqslant i \leqslant n}$ ou $(a_i)_{i \in J}$ ou encore (a_i), i.e. une suite finie d'éléments de E , où c(x) est le nombre de a_i égaux à x.

b.Def:

une partition d'une collection c est une collection p de sous-collections de c telle que $c = \sum_{c' \in p} c'$.

c.

Soient A et B deux collections de deux ensembles E et F.

c1.Définition (b-correspondance).

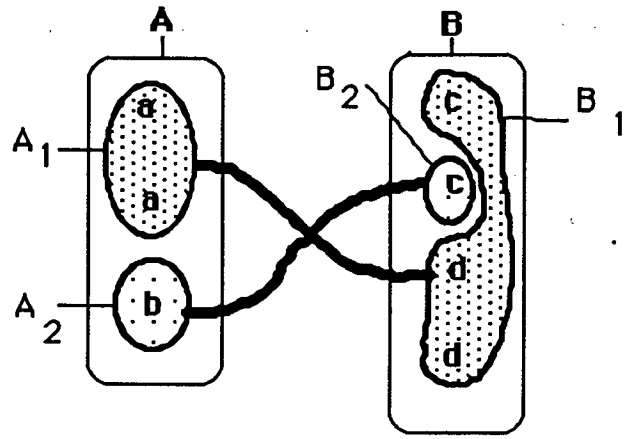
Une b-correspondance de A et B (b-corr en abrégé) est une collection de couples $((A_i, B_i))_{i \in J}$ tels que $(A_i)_{i \in J}$ et $(B_i)_{i \in J}$ sont des partitions de A et B respectivement.

Exemple:

$A = (a,a,b)$, $B = (c,c,d,d)$

$A_1 = (a,a)$ $A_2 = (b)$,

$B_1 = (c,d,d)$ $B_2 = (c)$



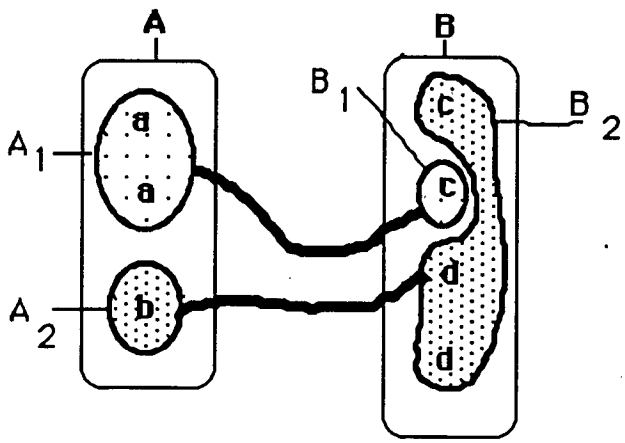
c2. Définition (a-correspondance):

soit $b = ((A_i,B_i))_{i\in J}$ une b-corr de A et B:

b est une a-corr de A et B $\Leftrightarrow \forall i\in J$, $\inf\{|A_i|,|B_i|\} = 1$

Exemple:

La b-correspondance de l'exemple c1. n'est pas une a-corr. En voici une:



c3. Définition et propriété:

On définit un ordre $\leqslant$ sur les b-corr de A et B par la fermeture transitive de la relation R suivante: $b\ R\ b' \Leftrightarrow |b-b'| = 1$. Si $b \leqslant b'$, alors b' peut être vu comme un raffinement de b.

Une a-correspondance de A et B est alors une b-corr maximale pour cet ordre.

La notion de a-correspondance est la base de notre algorithme de généralisation AC. Elle est d'une certaine manière nécessaire et suffisante à la généralisation AC comme on le montrera au §3.2.d.

d. Calcul de a-corr.

La détermination des a-corr de deux collections ne pose pas de problème particulier. On propose un algorithme en Annexe 4. Le nombre de a-corr est exponentiel en la taille des collections. Si on appelle na-corr(n,m) le nombre des a-corr de [1;n]et [1;m], on a:

na-corr(1,1) = 1

na-corr(2,2) = 2

na-corr(3,3) = 15

na-corr(4,4) = 184

na-corr(5,5) = 2945

...

na-corr(n,m) est plus grand que le nombre de surjections de [1;n]sur [1;m] ($n \geq m$), et plus petit que 2^{nm}.

3.2. Algorithme de généralisation AC.

a.Principe:

soient t et t' deux termes AC:

1. on construit récursivement un ensemble G(t,t') de minorants communs à t et t' qui a la propriété de contenir les généralisations principales de t et t'. Cela à partir des a-corr de t et t' et de leurs sous-termes.
2. Par des tests de filtrage AC on détermine les éléments maximaux de G(t,t') où on a auparavant éliminé des répétitions de termes égaux modulo renommage de variables.
3. Ces éléments maximaux constituent toutes las généralisations AC de t et t'.

b. Théorème de construction des généralisations AC .

Soit une bijection φ entre E(T) x E(T) et V.

Soient t_1 et t_2 deux termes AC.

On définit par induction l'ensemble $G(t_1,t_2)$:

- si $t_1 \in V$, ou $t_2 \in V$, ou t_1 et t_2 n'ont pas même racine, $G(t_1,t_2) = \{\varphi(t_1,t_2)\}$
- si $t_1 = ga_1...a_n$ et $t_2 = gb_1...b_n$ avec g non AC,

$$G(t_1,t_2) = \{gc_1...c_n \mid \forall i, c_i \in G(a_i, b_i)\}$$

- si $t_1 = fA$ et $t_2 = fB$ où f est AC,

$$G(t_1,t_2) = \bigcup_{((Ai,Bi))_{1\leqslant i\leqslant n} \text{ a-corr de A et B}} \left\{ f(c_1,...,c_n) \mid \forall i, c_i \in G(fA_i,fB_i) \right\}$$

Alors:

Les éléments maximaux pour $\leqslant_{ac}$ de $G(t_1, t_2)$ sont exactement les généralisations AC de t_1 et t_2.

De plus il existe des cas où les éléments de $G(t_1, t_2)$ sont exactement les généralisations AC de t_1 et t_2.

Afin de prouver ce théorème, nous aurons besoin de quelques propriétés .

c. Filtrage AC.

On notera fA un terme AC, où f est AC et A est une collection de termes AC. Si |A| = 1, fA désignera l'unique terme de A.

On a alors clairement:

Proposition:

soit $A = (a_i)_{i\in J}$, t= fA, t' = fB deux termes AC avec f AC,

alors

$\sigma t \simeq_{ac} t' \Leftrightarrow \exists (B_i)_{i\in J}$ une partition de B telle que $\forall i\in J, \sigma a_i \simeq_{ac} fB_i$.

d. Minorants communs et b-corr:

Soient $t_1 = fA$, $t_2 = fB$, $t = fC$, où f est AC, $C = (c_i)_{i\in J}$, alors

$t \leqslant_{ac} t_1$ et $t \leqslant_{ac} t_2 \Leftrightarrow \exists$ b une b-corr de A et B, $b = ((A_i,B_i))_{i\in J}$,

$\exists \sigma_1,\sigma_2$, telles que $\forall i\in J, \sigma_1 c_i \simeq_{ac} fA_i$, $\sigma_2 c_i \simeq_{ac} fB_i$

Ainsi, avec ces notations:

Proposition:

$renE(t) \in renE(t_1) \wedge renE(t_2) \Rightarrow$ b est une a-corr de A et B.

Preuve: supposons ue b n'est pas une a-corr de A et B: alors il existe 1, avec $|A_1| > 1$ et $|B_1| > 1$. Or la racine de c_1 ne peut être f, donc c_1 est une variable. Soient X et

Y deux nouvelles variables (n'apparaissant pas dans t , t_1 , t_2), x et y tels que A_1= (x) $\cup$ A'$_1$ et B_1= (y) $\cup$ B'$_1$, et d = fXY.

Alors $t <_{ac} t'$ = $fc_1 ... c_{1-1} d c_{1+1} ... c_n$, et $t' \leqslant_{ac} t_1$, $t' \leqslant_{ac} t_2$.

En effet si σ'_1 = {[X, x], [Y, fA'$_1$]} $\cup$ σ_1 et σ'_2 = {[X, y], [Y, fB'$_1$]} $\cup$ σ_2,

alors $\forall i=1,2$ $\sigma'_i t_i \simeq_{ac} t_i$.

Donc renE(t) n'est pas un minorant maximal de renE(t_1) et renE(t_2) .

CQFD.

Nous sommes maintenant en mesure de prouver le théorème 3.2.b.

Preuve du théorème 3.2.b:

Lemme:

(i) les éléments de G(t_1,t_2) sont des minorants de t $_1$et t_2 pour $\leqslant_{ac}$.

(ii) soit Min'(t_1,t_2) = { E(t") | $\overline{t"} = \overline{t'}_1 \wedge \overline{t'}_2$, $t'_1 \simeq_{ac} t_1$, $t'_2 \simeq_{ac} t_2$ }

alors $\forall$ t $\in$ Min'(t_1,t_2), $\exists \sigma$ plate, $\sigma t \in$ G(t_1,t_2)

Preuve du lemme:

(i) Soient σ_1 et σ_2 définies par $\forall a,b \in E(T)$, σ_1 (φ(a,b)) = a, σ_2(φ(a,b)) = b.

Montrons par induction que $\forall t \in G(t_1,t_2)$, $\sigma_1 t \simeq_{ac} t_1$ et $\sigma_2 t \simeq_{ac} t_2$.

- si $t_1 \in V$, ou $t_2 \in V$, ou t_1 et t_2 n'ont pas même racine, c'est clair car G(t_1,t_2) = {φ(t_1,t_2)}

- si $t_1 = ga_1 ... a_n$ et $t_2 = gb_1 ... b_n$ avec g non AC:

soit t$\in$ G(t_1,t_2): $t = gc_1 ... c_n$, $\forall i$ $c_i \in G(a_i,b_i)$

par hypothèse d'induction on a $\forall i$, $\sigma_1 c_i \simeq_{ac} a_i$, $\sigma_2 c_i \simeq_{ac} b_i$ donc aussi

$\sigma_1 t \simeq_{ac} t_1$ et $\sigma_2 t \simeq_{ac} t_2$, cqfd.

- si t_1 = fA et t_2 = fB, avec f AC:

$t \in G(t_1,t_2)$ est de la forme $f(c_1,...,c_n)$ où $\forall i$ $c_i \in G(fA_i,fB_i)$ (cf définition de G(t,t')). Par hypothèse d'induction on a $\forall i$, $\sigma_1 c_i \simeq_{ac} fA_i$, $\sigma_2 c_i \simeq_{ac} fB_i$ donc aussi $\sigma_1 t \simeq_{ac} t_1$ et $\sigma_2 t \simeq_{ac} t_2$, cqfd.

Ainsi les éléments de G(t_1,t_2) sont des minorants de t_1 et t_2 pour $\leqslant_{ac}$.

(ii) on montre par induction l'hypothèse suivante:

E(t)$\in$Min'(t_1,t_2), $\overline{t} = \overline{t'}_1 \wedge \overline{t'}_2$, $t'_1 \simeq_{ac} t_1$, $t'_2 \simeq_{ac} t_2$, $\sigma_1 t = t'_1$, $\sigma_2 t = t'_2$

$\Rightarrow$ si σ est définie par $\sigma(X) = \varphi(\sigma_1(X),\sigma_2(X))$ alors $E(\sigma t) \in G(t_1,t_2)$

- si t_1 et t_2n'ont pas même racine ou si l'un des deux est une variable, Min'(t_1,t_2)={E(X)}, donc $\sigma(t) = \varphi(t_1,t_2) \in G(t_1,t_2)$, CQFD.

- si $t_1 = ga_1 ... a_n$ et $t_2 = gb_1 ... b_n$ avec g non AC:

alors $t = g\, c_1 ... c_n$, $\sigma_1 t = g\, a'_1 ... a'_n$, $\sigma_2 t = g\, b'_1 ... b'_n$, avec $\forall i$, $a'_i \simeq_{ac} a_i$, $b'_i \simeq_{ac} b_i$.

Alors $\forall i, \overline{c_i} = \overline{a'_i} \wedge \overline{b'_i}, \sigma_1 c_i = a'_i, \sigma_2 c_i = b'_i$, donc par hypothèse d'induction, on a $E(\sigma c_i) \in G(a_i,b_i)$, et donc $E(\sigma t) \in G(t_1,t_2)$, CQFD.

- si $t_1 = fA$ et $t_2 = fB$, avec f AC:
Il est clair qu'il existe une a-corr $((A_i,B_i))_{1 \leqslant i \leqslant n}$ de A et B telle le terme t de l'hypothèse vérifie:
$E(t) = E(f(c_1,...f(c_{n-1},c_n)...))$, avec $\forall i, E(c_i) \in Min'(fA_i, fB_i), E(\sigma_1 c_i) = fA_i, E(\sigma_2 c_i) = fB_i$,. Par hypothèse d'induction, $\forall i, E(\sigma c_i) \in G(a_i,b_i)$, et donc $E(\sigma t) \in G(t_1,t_2)$, CQFD.

Le lemme est donc démontré.

Revenons au théorème:
comme Min'(t,t') contient toutes les généralisations principales de t et t' (propriété 1.4.a.) alors G(t,t') aussi, et le premier résultat en découle .
L'exemple suivant illustre le deuxième résultat.
CQFD.

e. Un exemple montrant la nécessité du calcul de toutes les a-corr pour la généralisation AC:

Cet exemple montre que d'une certaine manière la notion de a-correspondance est nécessaire et suffisante à la généralisation AC.
Soient fA et fB deux termes AC, où f est AC, A et B deux collections de termes de racines différente de f, les termes de A ayant des racines différentes de celles des termes de B.
Soit $ACORR(A,B) = \{ ((A_{ji},B_{ji}))_{1 \leqslant i \leqslant k_j} , 1 \leqslant j \leqslant n\}$ l'ensemble de toutes les a-corr de A et B.

Soit g une fonction d'arité 1 + \i\su(b∈ACORR(A,B];; |b|).
Soient $t = g(fA, fA_{11},..., fA_{1k_1}, ... , fA_{n1},..., fA_{nk_n})$
et $t' = g(fB, fB_{11},..., fB_{1k_1}, ... , fB_{n1},..., fB_{nk_n})$
Soient $\forall j,i, X_{ji} = \varphi(fA_{ji}, fB_{ji})$
et $\forall r, 1 \leqslant r \leqslant n, t_r = g(f X_{r1} ... X_{rk_r}, X_{11},..., X_{1k_1}, ... , X_{n1},..., X_{nk_n})$
alors les $renE(t_r)$ sont distinctes deux à deux et constituent exactement les n généralisations principales de renE(t) et renE(t'). On a de plus l'égalité $\#G(t,t') = n$ ce qui montre que dans ce cas, G(t,t') ne contient que les généralisations AC de t et t', et donc que toutes les a-corr de A et B sont nécessaires.

En annexe on trouvera des exemples de calculs de généralisations AC avec l'implémentation de l'algorithme décrit. Les exemples de ce paragraphe y sont traités.

On remarque enfin qu'expérimentalement ce n'est pas la construction de G(t,t') qui prend du temps, malgré sa taille qui peut être importante (mais qui se réduit considérablement lorqu'on y élimine les redondances par renommage), mais c'est plutôt le calcul de ses éléments maximaux, et donc le filtrage AC.

3.3. Généralisation AC d'un nombre fini de termes.

Il s'agit de déterminer les éléments maximaux parmi les minorants communs à $renE(t_1),\ldots,renE(t_n)$. Leur ensemble est noté $\bigwedge_{1\leqslant i\leqslant n} renE(t_i)$.

a. Propriété:

$\bigwedge_{1\leqslant i\leqslant n} renE(t_i)$ est constitué des éléments maximaux de

$$\bigcup_{renE(t') \in \bigwedge_{1\leqslant i\leqslant n-1} renE(t_i)} renE(t') \wedge renE(t_n)$$

Cette propriété est immédiate, et il en découle clairement un algorithme récursif de calcul de $\bigwedge_{1\leqslant i\leqslant n} renE(t_i)$, en utilisant l'algorithme de généralisation AC de deux termes.

3.4. Lien avec le système SGAC.

L'algorithme présenté correspond à une stratégie particulière d'application des règles de SGAC:

Le travail de la bijection du théorème 3.2.b. peut être fait par la règle $RGAC_3$, qui doit être alors appliquée en priorité.

Les règles $RGAC_1$ et $RGAC_2$ construisent en fait des b-correspondances croissantes, donc au final des a-correspondances, qui donneront les généralisations AC principales.

Ces a-correspondances sont construites par l'algorithme construisant l'ensemble G(t,t'), qui est alors l'ensemble des formes normales par SGAC du problème initial.

Il reste à tester sur une implémentation de SGAC, si l'avantage qu'il a de factoriser des problèmes de généralisation répétés , compense son désavantage qui est de construire toutes les b-correspondances intermédiaires, alors que l'algorithme présenté ne manipule que des a-correspondances.

Toutefois la règles $RGAC_1$' suivante peut peut-être remplacer avantageusement $RGAC_1$, puisqu'elle ne construit que des a-correspondances (le nouveau système SGAC'est équivalent à SGAC, puisque $RGAC_1$' compacte clairement les itérations de $RGAC_1$):

$$RGAC_1' : \quad \frac{G \mid fA =_x fB, S}{\{[x \to fy_1 \ldots y_n]\}G \mid fA_1 =_{y1} fB_1, \ldots, fA_n =_{yn} fB_n, S}$$

où f est AC, $y_1, \ldots, y_n$ de nouvelles variables, et $((A_i, B_i))_{1 \leqslant i \leqslant n}$ une a-correspondance de A et B.

Dans ce cas l'algorithme présenté devient très proche du système SGAC' utilisé avec la stratégie qui consiste à favoriser la règle $RGAC_3$.
La encore une implémentation de ce système devrait déterminer dans quelle mesure l'algorithme présenté améliore le système SGAC' (il est à noter que cet algorithme évite la complexité du filtrage nécessaire à l'application des règles d'inférence).

4. GENERALISATION AC: CAS LINEAIRE .

On s'intéresse ici aux généralisations AC linéaires principales de deux termes renE(t) et renE(t'), i.e. aux éléments maximaux de l'ensemble des minorants linéaires de ces termes. La linéarité est stable par $\simeq_{ac}$ et par renommage, on peut donc la définir naturellement sur renE(T). On notera $renE(t) \wedge_{lin} renE(t')$ leur ensemble et par abus de langage on les appellera "généralisations AC linéaires de t et t' ". Elles sont en nombre fini non nul, puisque deux termes de renE(T) ont un nombre fini non nul de minorants communs.

Le résultat suivant ne pose pas de difficulté:

4.1.Théorème:

$renE(t) \in renE(t_1) \wedge_{lin} renE(t_2) \Rightarrow \exists \sigma, \sigma \text{ plate}, renE(\sigma t) \in renE(t_1) \wedge renE(t_2)$

Corollaire:

$\# renE(t_1) \wedge_{lin} renE(t_2) \leqslant \# renE(t_1) \wedge renE(t_2)$

Les généralisations AC linéaires ne diffèrent donc des généralisations AC que par leurs variables, et sont moins nombreuses que celles-ci, ce qui rend intéressant leur calcul, à partir du moment où il est plus simple, comme on le verra dans la suite.

On propose un algorithme de calcul de ces généralisations linéaires, proche de l'algorithme de 3., et dont on montrera qu'il peut être vu comme une utilisation du système SGAC réduit à ses deux premières règles.

Pour calculer ces généralisation AC linéaires, nous aurons besoin d'une nouvelle notion, celle de "c-correspondance" de collections de termes AC.

4.2. c-correspondances.

Soient A et B deux collections de termes AC.

a. Définition.

Soit $b = ((A_i,B_i))_{1 \leqslant i \leqslant n}$ une b-correspondance de A et B.

La partie unitaire de b, notée U(b), est la sous-collection de b constituée des couples où $|A_i|=|B_i|=1$, les deux uniques termes de A_i et B_i n'étant pas des variables, et ayant la même racine.

b. Définition:

Une c-correspondance (c-corr en abrégé) est la partie unitaire d'une b-corr maximale pour le pré-ordre défini par $b \leqslant b' \Leftrightarrow U(b) \subset U(b')$.

Si $A = (a_i)_{1 \leqslant i \leqslant p}$ et $B = (b_j)_{1 \leqslant j \leqslant q}$ on notera une c-corr par $c = ((a_{i_k}, b_{i_k}))_{1 \leqslant k \leqslant n}$. On a alors $\forall k, a_{i_k} \notin V$, $b_{i_k} \notin V$, a_{i_k} et b_{i_k} ont même racine.

c. Exemples:

1.

A = (a, X, f(a), f(e)), B = (a, Y, Z, f(d), f(e), g(a,a)):

c = ((a,a), (f(a), f(d)), (f(e), f(e))) est une c-corr de A et B: c'est la partie unitaire de la b-corr b = (((a),(a)) , ((f(a)),(f(d))) , ((f(e)),(f(e))) , ((X),(Y, Z, g(a,a)))).

c' = ((a,a), (f(a), f(d))) est la partie unitaire de

b' = (((a),(a)) , ((f(a)),(f(d))) , ((f(e), X), (Y, Z, f(e), g(a,a)))) mais n'est pas une c-corr, car elle est une sous-collection stricte de c.

2.

A = (a, b,b), B = (a, b):

c_1 = ((a,a)) et c_2 = ((b,b)) sont les deux seules c-corr de A et B.

c = ((a,a) , (b,b)) n'est pas une c-corr de A et B car c ne peut être une sous-collection d'une b-corr de A et B.

On donne en annexe 4 un algorithme de calcul des c-correspondances de deux collections de termes AC.

4.3. Algorithme de calcul des généralisations AC linéaires.

a. Définition:

Soit $E = (a_1, \ldots, a_n)$ une collection de termes AC. On définit récursivement MAX(E) par:

si n=0,

alors MAX(E) = $\varnothing$

sinon $MAX(E) = (a_1) \cup (E - \{a_i \mid i \geqslant 2, renE(a_i) = renE(a_1)\})$

MAX(E) est alors un ensemble minimal de représentants des éléments maximaux de E pour le pré-ordre $\leqslant_{ac}$.

b. Théorème (construction des généralisations AC linéaires):

Soit NV une fonction sans argument qui retourne à chaque appel une variable nouvelle (par incrémentation d'un compteur par exemple).

Soient t_1 et t_2 deux termes AC.

On définit par induction l'ensemble $GL(t_1,t_2)$:

- si $t_1 \in V$, ou $t_2 \in V$, ou t_1 et t_2 n'ont pas même racine, $GL(t_1,t_2) = \{NV\}$
- si $t_1 = ga_1...a_n$ et $t_2 = gb_1...b_n$ avec g non AC,

$$GL(t_1,t_2) = \{gc_1...c_n \mid \forall i, c_i \in GL(a_i, b_i)\}$$

- si $t_1 = fA$ et $t_2 = fB$ où f est AC,

$$GL(t_1,t_2) = MAX\Big(\bigcup_{((ai,bi))_{1\leq i\leq n} \text{ c-corr de A et B}} \{ f(c_1,...,c_n, NV,...,NV) \mid \forall i, c_i \in GL(fa_i,fb_i) \} \Big)$$

où NV est répétée r fois , $r = \inf(|A|,|B|) - n$.

Soit $renE(GL(t_1,t_2)) = \{renE(t) \mid t \in GL(t_1, t_2)\}$.

Alors $\#renE(GL(t_1,t_2)) = \# GL(t_1, t_2)$

et $renE(GL(t_1,t_2))$ est l'ensemble des éléments maximaux de l'ensemble des minorants linéaires de $renE(t_1)$ et $renE(t_2)$. C'est l'ensemble des généralisations AC linéaires de t_1 et t_2.

Preuve: elle est semblable à celle du théorème 3.2.b.

Il résulte de ce théorème un algorithme de construction de GL(t,t'), qu'on a implémenté, et dont des exemples d'éxécution sont donnés en annexe.

4.4. Complexité de la généralisation AC linéaire:

Théorème:

Le nombre de généralisations AC linéaires est exponentiel en la taille des termes de départ.

Preuve:

La majoration exponentielle résulte du théorème similaire sur la généralisation AC (1.4.c.) .

L'exemple suivant donne une minoration exponentielle:

soient $t_1 = f\,ab$ et $t_2 = f\,abb$, et $g_0 = f\,aX$, $g_1 = f\,bY$ leurs deux généralisations AC linéaires.

soient h une fonction n-aire, $t'_1 = h(t_1,...,t_1)$ et $t'_2 = h(t_2,...,t_2)$. Alors t'_1 et t'_2 ont 2^n généralisations AC linéaires qui sont les termes de la forme:

$h(f\,c_1X_1, ..., f\,c_nX_n)$, où $\forall i, c_i \in \{a,b\}$.

CQFD.

4.5. Comparaison avec la généralisation AC:

Il y a beaucoup moins de c-corr que de a-corr, le filtrage AC est plus rapide sur des termes linéaires, ce sont deux raisons qui expliquent que cet algorithme de généralisation AC linéaire est beaucoup plus rapide que l'algorithme de généralisation AC du §3. . Les exemples de généralisations AC linéaires donnés en annexe, qui incluent ceux de la généralisation AC, le montrent clairement.

4.6. Généralisations AC linéaires d'un nombre fini de termes.

On suit la même méthode qu'en 3.3.:

Il s'agit de déterminer les éléments maximaux parmi les minorants linéaires communs à $renE(t_1),\ldots,renE(t_n)$. Leur ensemble est noté $\bigwedge_{1\leqslant i\leqslant n}^{lin} renE(t_i)$.

a. Propriété:

$\bigwedge_{1\leqslant i\leqslant n}^{lin} renE(t_i)$ est constitué des éléments maximaux de

$$\bigcup_{renE(t') \in \bigwedge_{1\leqslant i\leqslant n-1}^{lin} renE(t_i)} renE(t') \wedge_{lin} renE(t_n)$$

Cette propriété est immédiate, et il en découle clairement un algorithme récursif de calcul de $\bigwedge_{1\leqslant i\leqslant n}^{lin} renE(t_i)$, en utilisant l'algorithme de généralisation AC linéaire de deux termes.

4.7. Lien avec le système SGAC:

Il est clair que le système SGAC restrient aux règles $RGAC_1$ et $RGAC_2$ (appelons le SGAClin) produit des généralisations AC linéaires de deux termes. Les lemmes de 2. permettent alors de montrer que parmi ces généralisations linéaires se trouvent les généralisations AC linéaires principales. L'algorithme précédent apparaît alors très proche du système SGAClin, comme l'algorithme de 3. pour le système SGAC',.

5. GENERALISATION DE TERMES fx^u où f est AC et x est un vecteur de variables.

5.1. Réduction du problème aux partitions d'entiers.

a. Définition:

Une partition d'entier est une suite décroissante d'entiers non nuls . Si $u = (u_1,\dots,u_p)$ est une partition d'entier, on dit que c'est une partition de $n = \sum u_i = u_1 + \dots + u_p$. n est alors la taille de u, notée $|u|$, et sa longueur est l'entier p.

b. L'ordre $\leqslant$ sur les partitions d'entiers.

Soient $u = (u_1,\dots,u_p)$ et $v = (v_1,\dots,v_q)$ deux partitions d'entiers.

On définit $u \leqslant v \iff \exists L \in M_{q,p}(\mathbb{N})$ sans colonne nulle, $Lu=v$.

Proposition:

$\leqslant$ est une relation d'ordre, et les minorants d'une partition sont en nombre fini. On note alors $u \wedge v$ l'ensemble des minorants maximaux de u et v (qui sont donc en nombre fini , et non nul puisque $\forall u, (1) \leqslant u$).

c. Propriété de réduction:

Soit x et y des vecteurs de variables distinctes , u et v deux partitions d'entiers:

$renE(fz^w) \in renE(fx^u) \wedge renE(fy^v) \iff w \in u \wedge v$ et z est un vecteur de variables toutes distinctes.

(On a juste reformulé le théorème 3 du §1.4.d2.)

On est donc ramené au calcul de $u \wedge v$, ce qui nous conduit à l'étude des dénumérants et des semi-groupes de $(\mathbb{N}^*,+)$, étude que l'on va mener au paragraphe suivant, et qu'on utilisera ensuite pour donner deux algorithmes, l'un décidant l'ordre $\leqslant$, l'autre calculant $u \wedge v$.

5.2. Semi-groupes de $(\mathbb{N}^*,+)$.

Soit $u \in \mathbb{N}^{*p}$, $u = (u_1,\dots,u_p)$. On note (u) le semi-groupe de $(\mathbb{N}^*,+)$ engendré par les u_i. (i.e. les combinaisons linéaires à coefficients entiers naturels des u_i).

On note $\mathbb{1}(u)$ le complémentaire de (u) dans $\mathbb{N}^*$.

On définit une relation de préordre $\leqslant_1$ par

$u \leqslant_1 v \iff \forall j, v_j \in (u)$

On a alors $u \leqslant_1 v \iff \exists L \in M_{q,p}(\mathbb{N})$, $Lu=v$, et donc $u \leqslant v \Rightarrow u \leqslant_1 v$.

Calculer $u \wedge v$ nécessite la décision de l'ordre $\leqslant$, qui a comme condition nécessaire le préordre $\leqslant_1$, et donc conduit à tester l'appartenance à un semi-groupe de $\mathbb{N}^*$.

On propose une compilation du test $n \in (u)$, après avoir énoncé quelques propriétés des semi-groupes de $\mathbb{N}^*$.

a.Proposition et définition:

soit $\wedge u$ le pgcd des $u_1, \ldots, u_p$:

$\wedge u = 1 \Rightarrow l(u)$ est fini.

Dans ce cas on appelle "conducteur de u" l'entier $c(u) = \sup(l(u)) + 1$.

L'étude du conducteur fait l'objet du "problème de Frobenius".

b. Bornes du conducteur:

Les énoncés suivants se montrent sans difficulté:

soit u tel que $\wedge u = 1$, alors:

- $c(u) \leqslant (u_i - 1) \sum_{j \neq i} u_j$

- si u_i et u_j sont premiers entre eux, $c(u) \leqslant (u_i - 1)\, u_j$.

- $\exists J \subset [1;p]$ tel que $(u) = (u_j)_{j \subset J}$, et cardinal$(J) \leqslant \inf \{u_i,\ 1 \leqslant i \leqslant p\}$; dans ce cas on a, si $k \in J$, $c(u) \leqslant (u_k - 1) \sum_{j \in J - \{k\}} u_j$

- si n et m sont premiers entre eux, $c((n,m)) = (n-1)(m-1)$: les bornes précédentes sont donc quasiment atteintes.

- $c(u)/2 \leqslant \#l(u) \leqslant c(u) - 1$: le nombre d'éléments de l(u) est donc de l'ordre de c(u).

J. Dixmier a récemment démontré et affiné une conjecture d'Erdös et Graham (donnant en outre d'autres résultats sur le conducteur dans des cas particuliers):

Théorème [Dixmier 88]:

Soient $2 \leqslant b < a$, et g(a,b) le plus grand des conducteurs des semi-groupes $(u_1, \ldots, u_b)$ où les u_i sont compris entre 1 et a.

Alors $[(a-2)/(b-1)]\,(a-b+1) - 1 \leqslant g(a,b) \leqslant (\{(a-1)/(b-1)\} - 1)\, a - 1$.

(où [x] est la partie entière de x et {x} le plus petit entier supérieur à x).

Enfin, on a immédiatement, concernant l(u), la propriété suivante:

c.Prop:

(on note $a \mid b$ si a divise b)

$n \in (u) \Leftrightarrow \wedge u \mid n$ et $n/\wedge u \notin l(u/\wedge u)$,

où $u/\wedge u$ dénote $(u_1/\wedge u, \ldots, u_p/\wedge u)$.

Si on connaît $l(u/\wedge u)$ on peut donc tester rapidement $n \in (u)$.

d. Structure et calcul de l(u):

soit $u = (u_1,\dots,u_p)$, $\wedge u = 1$, $u' = (u_1,\dots,u_{p-1})$:

Théorème (structure de l(u)):

$$l(u) = \bigcup_{x \in F} E(x,k(x))$$

où $F = \{x \mid 1 \leqslant x < u_p, x \notin (u')\}$

si $x \in F$, $k(x) = \sup\{k \mid k' \leqslant k \Rightarrow x + k'u_p \notin (u')\}$

et $E(x,k) = \{x + k'u_p, \text{ où } 0 \leqslant k' \leqslant k\}$

Preuve: elle repose sur le lemme suivant dont la preuve est sans difficulté:

Lemme :

$\forall y>0,\ y + u_p \in l(u) \Leftrightarrow y \in l(u)$ et $y+u_p \notin (u')$.

CQFD.

On a donc un grande régularité dans la structure de l(u), et l(u) est déterminée par F (au plus u_p-1 éléments), et les k(x) pour chaque $x \in F$. Donc $2u_p - 2$ entiers au plus caractérisent l(u).

Par exemple :

$u = (7, 5, 4)$, $u_p = 4$

$F = \{1, 2, 3\}$, $k(1)= 0$, $k(2)= 1$, $k(3)= 0$.

$l(u) = \{1, 2, 3, 6\}$

$c(u) = 7$

la structure de l(u) s'illustre bien sur un dessin :

. . . x x . x x x x ...

1 2 3 4 5 6 7 8 9 10 ...

e. Test $n \in (u)$:

Avec ces notations on a alors:

Corollaire (test $n \in (u)$):

Soient $u = (u_1, \dots, u_p)$ et $l(u/\wedge u)$ représentée par F et les k(x) pour chaque x de F.

Alors:

$n \in (u) \Leftrightarrow \wedge u \mid n$

et $\forall x \in F,\ \frac{n}{\wedge u} - x = \frac{k\,u_p}{\wedge u} \Rightarrow k > k(x)$

Le test $n \in (u)$ est donc très rapide, sa complexité ne dépendant essentiellement que du cardinal de F, i.e. de $\frac{u_p}{\wedge u}$. La compilation de ce test consiste à déterminer F et les k(x), et nécessite

la compilation du test d'appartenance à u' = $(u_1, ..., u_{p-1})$. Ainsi cette compilation est récursive sur u.

f. Evaluation de la méthode, comparaisons.

On a implémentée cette méthode en Le_Lisp sur le Gould de l'INRIA Sophia Antipolis. Des tests sur des vecteurs aléatoires de coefficients inférieurs à 100 et de longueur inférieure à 30 donnent des temps d'éxécution inférieurs à la seconde pour la compilation du test, et inférieurs à 2.10^{-3}s pour le test lui-même.

On a aussi implémenté la méthode qui consiste à résoudre $n \in (u)$ en testant l'existence de solution pour l'équation $n = x_1u_1 + ... + x_pu_p$, i.e.

$n \in (u) \Leftrightarrow \wedge u | n$

et $\exists x_1 \in [0; n/u_1]$, tel que $n - x_1u_1 \in ((u_2,...,u_p))$

Les x_i sont testés en décroissant et les u_i sont supposés croissants.

Toujours pour des u vérifiant $\forall i$, $u_i \leqslant 100$ et $p \leqslant 30$, on a alors des temps d'éxécution du test $n \in (u)$ de l'ordre de 5.10^{-2}s (la taille des n n'intervient quasiment pas, comme pour la méthode précédente).

Voici un tableau comparatif des deux méthodes:

t_1 est le temps moyen (en seconde) du test $n \in (u)$ avec notre méthode

t_2 celui avec la résolution de l'équation diophantienne,

t_c celui de la compilation du test $n \in (u)$.

	t_c	t_1	t_2	t_2/t_1
$u_i \leqslant 10, p \leqslant 6$	$<10^{-6}$	2.10^{-4}	14.10^{-4}	7
$u_i \leqslant 30, p \leqslant 15$	2.10^{-2}	$3,5.10^{-4}$	8.10^{-3}	22
$u_i \leqslant 100, p \leqslant 30$	0,5	$1,4.10^{-3}$	$6,8.10^{-2}$	50
$u_i \leqslant 500, p \leqslant 50$	20	6.10^{-3}	0,6	100

Trois exemples de calcul de l(u) sont donnés en annexe.

5.3. Test de $u \leqslant v$:

Le problème est de trouver une matrice L sans colonne nulle telle que Lu=v. Les conditions nécessaires suivantes permettent de conclure dans de nombreux cas:

a. Conditions nécessaires:

Soient u et v deux partitions d'entiers, alors $u \leqslant v \Rightarrow$

a1 $\forall p \in \mathbb{N}^*$, $\sum u_i^p \leqslant \sum v_j^p$

a2 en particulier p=1: $|u| \leqslant |v|$

a3 $p = \infty$: $\sup(u) \leqslant \sup(v)$

a4 $\inf(u) \leqslant \inf(v)$

a5 $u \leqslant_{lex} v$, où $\leqslant_{lex}$ est l'ordre lexicographique.

a6 $u \leqslant_1 v$ (i.e. $\forall v_{j_i}, v_j \in (u)$)

(résolu par les tests d'appartenance des v_j à (u), d'où l'intérêt de la compilation du test $n \in (u)$).

a7 $\wedge u$ divise $\wedge v$

b. Algorithme:

Soient $u = (u_1, \ldots, u_p)$ et $v = (v_1, \ldots, v_q)$ deux partitions d'entiers. Tester $u \leqslant v$ va revenir à "construire" les v_j par somme de coefficients de u (i.e. $Lu=v$), en utilisant chaque u_i au moins une fois (i.e. L sans colonne nulle). On va garder à chaque étape un ensemble I d'indices correspondant aux u_i restant à utiliser pour construire v.

$u \leqslant v$:

soit F_u la fonction testant l'appartenance à (u), provenant de la compilation de ce test par la méthode exposée en 2.2.

si une des conditions a2, ..., a7 n'est pas vérifiée,

alors rendre FAUX

sinon rendre TEST(v, [1;p])

La fonction TEST(v,L) est la suivante:

TEST(v',L) =

si $L=\varnothing$

alors $(\forall j, F_u(v'_j)$ ou $v'_j = 0)$

sinon soit $i \in I$:

$\exists j, v'_j \geqslant u_i$ et $TEST((v'_1, \ldots, v'_{j-1}, v'_j - u_i, v'_{j+1}, \ldots, v'_q), L-\{i\})$

Remarquons que la compilation du test d'appartenance à (u) est faite une fois, et que ce test est utilisé de nombreuses fois par la suite.

Voici un tableau comparant cet algorithme et sa variante où le test $n \in (u)$ est effectué par la méthode de l'équation diophantienne (cf §5.2.f.):

t_1 donne le temps moyen du test $u \leqslant v$ avec l'algorithme décrit,

t_2 celui de la méthode utilisant l'équation diophantienne.

$u \leqslant v$	t_1	t_2	t_1/t_2
$u_i, v_j \leqslant 9,\ p,q \leqslant 5$	7.10^{-3}	14.10^{-3}	2
$u_i, v_j \leqslant 3,\ p,q \leqslant 3$	4.10^{-3}	$7{,}5.10^{-3}$	$\simeq 2$
$u_i, v_j \leqslant 19,\ p,q \leqslant 9$	$2{,}7.10^{-2}$	$3{,}7.10^{-2}$	1,7

Notre méthode de compilation du test $n \in (u)$ est donc avantageuse.

5.4. Calcul rapide d'un élément de $u \wedge v$:

Soient $u = (u_1,\ldots,u_p)$ et $v = (v_1,\ldots,v_q)$ deux partitions d'entiers. On cherche ici non pas tous les éléments de $u \wedge v$ mais un seul. L'algorithme qu'on propose ici est rapide et utilise un algorithme très rapide qui donne un minorant de u et v, qui est en pratique souvent maximum, et sinon "presque maximum".

a. Un minorant "presque maximum":

La fonction suivante MINOR(u,v) donne un minorant w de u et v vérifiant $|w| = \inf\{|u|,|v|\}$. Pour simplifier, disons que les coefficients de MINOR(u,v) sont obtenus en prenant à chaque étape le plus petit des coefficients maximum de u et v, et en le soustrayant à ceux-ci. Pour définir MINOR(u,v), on utilise une fonction TRIE qui ordonne une liste d'entiers en décroissant en éliminant les nuls:

MINOR(u,v)=
si p=1
 alors si $u_1=1$
  alors (1)
  sinon si q=1
   alors si $v_1=1$
    alors (1)
    sinon (inf{u_1-1,v_1-1}, 1)
   sinon si $u_1 \leqslant v_1$
    alors (u_1-1,1)
    sinon (v_1,w_2,...,w_k)
     où (w_2,...,w_k):= MINOR((u_1-v_1),(v_2,...,v_q))
sinon si q=1
 alors MINOR(v,u)
 sinon (w_1,w_2,...,w_k)
  où w_1= inf{u_1,v_1}
   u'= TRIE (u_1-w_1,u_2,...,u_p)
   v'= TRIE (v_1-w_1,v_2,...,v_p)
   (w_2,...,w_k) = MINOR(u',v')

b. Un élément de u∧v:

Définition:

w' est dit "successeur immédiat" de w ssi w< w" ⩽ w' ⇒ w"= w'.

Définition et proposition:

On définit u $\wedge_1$ v algorithmiquement:

(0) soit w = MINOR(u,v)
(1) s'il existe w' sucesseur immédiat de w de même taille que w, qui vérifie w'⩽u et w'⩽v,
  alors aller en (1) avec w := w',
  sinon u $\wedge_1$ v := w'.

Alors u$\wedge_1$v ∈ u∧v.

Preuve:

Lemme:

soit $w = (w_1,\dots,w_k)$, $w' = (w'_1,\dots,w'_{k'})$, $w' > w$, $|w'| = |w|$,

alors $\exists w''$, $w < w'' \leqslant w'$,

$\exists\, i,j\ 1 \leqslant i < j \leqslant k$ tels que

$w'' = \text{TRIE}(w_i+w_j, w_1,\dots,w_{i-1},w_{i+1},\dots,w_{j-1},w_{j+1},\dots,w_k)$

(La fonction TRIE est la même qu'en a.)

Ce lemme est clair car w'' est obtenu par regroupement de deux coefficients de w.

Il est donc facile de déterminer les successeurs immédiats de même taille d'une partition d'entier.

En remarquant que les minorants d'une partition sont en nombre fini, il est clair que $u \wedge_1 v$ est bien un minorant maximum de u et v.

CQFD.

Des tests sur des partitions d'entiers de coefficients inférieurs à 100 et avec au plus 30 coefficients ont montré que l'étape (1) est très peu itérée (1 à 3 fois). Le calcul d'un minorant maximum est alors très rapide.

5.5. Calcul de $u \wedge v$.

On sait que $w \leqslant u \Rightarrow |w| \leqslant |u|$ et $w \leqslant_1 u$ ($\leqslant_1$ est l'ordre lexicographique). Les minorants communs à u et v seront donc à chercher parmi les partitions de taille inférieure à celles de u et v, et de plus lexicographiquement inférieures à u et v. L'idée de l'algorithme qu'on propose est d'énumérer ces partitions tout en gardant au fur et à mesure les partitions maximales parmi celles déjà énumérées.

a. Définitions:

On définit l'ordre total suivant:

$u \leqslant_2 v \iff |u| < |v|$ ou ($|u| = |v|$ et $u \leqslant_{lex} v$),

la fonction qui donne le prédécesseur d'une partition pour cet ordre:
$\text{PRED}((1)) = \varnothing$ et $\text{PRED}(w) = \sup_{\leqslant_2} \{ w' \mid w' <_2 w \}$,

et la fonction SAUTS définie par:
$\text{SAUTS}(w) = \sup_{\leqslant_2} \{w' \mid w' \leqslant_2 w \text{ et } \text{non}(w' \leqslant w)\}$

b. Algorithme:

Soit u et v deux partitions d'entiers.

On énumère en décroissant pour $\leqslant_2$ les partitions inférieures à u et à v pour $\leqslant_2$. Pour cela on utilise la fonction PRED . On utilise la fonction SAUTN pour de plus éviter certaines partitions dont on sait alors qu'elle ne sont pas dans $u \wedge v$. L'ensemble LM contient les partitions maximales parmi celles déjà rencontées.

$u \wedge v$ =
soit $w = \sup_{\leqslant 2}\{w' \mid w' \leqslant_2 u, w' \leqslant_2 v\}$
soit LM = $\varnothing$
tant que $w \neq \varnothing$
    si $\forall w' \in$ LM, non($w \leqslant w'$)
    alors si $w \leqslant u$ et $w \leqslant v$
        alors LM := $\{w\} \cup \{w' \in$ LM, non($w' \leqslant w$)$\}$
            w := SAUTS(w)
        sinon w := PRED(w)
    sinon w := SAUTS(w)
$u \wedge v$:= LM

La fonction SAUTS calculée ainsi (on utilise une fonction accélératrice SAUT1 itérée):

SAUT1(w) =
si $w = (w_1,\ldots,w_k)$ avec $w_k > 1$
    alors PRED($(w_1,\ldots,w_{k-1},1,\ldots,1)$) - où 1 apparaît w_k fois -
    sinon w

SAUTS(w) =
soit $w = (w_1,\ldots,w_k)$
soit w' = (si $w_k > 1$ alors SAUT1(w) sinon PRED(w))
si $w' \leqslant w$ et $w' \neq \varnothing$
    alors SAUTS(w')
    sinon w'

Théorème:

Cet algorithme rend l'ensemble $u \wedge v$ des minorants maximaux de u et v pour l'ordre $\leqslant$.

Preuve:

Le seul point à préciser est la fonction SAUT1, dont l'intérêt provient du lemme suivant:

Lemme:

soit $w \leqslant u$, $w \leqslant v$, $w = (w_1,\ldots,w_k)$, $w_k > 1$,
il existe alors $w' = (w_1,\ldots,w_{k-1},w'_1,\ldots,w'_1)$ avec $1 \neq 1$, $|w'| = |w|$
on a $w < w'$ et donc w n'est pas un minorant maximum de u et v.

CQFD.

Des exemples de calculs de $u \wedge_1 v$ et de $u \wedge v$ sont donnés en annexe.

E. CONCLUSION.

Comme le montrent les exemples traités en annexe, les algorithmes de généralisation AC proposés sont efficaces, et rendent praticable la recherche de généralisations AC, ce qui était impossible de manière naïve. Concernant d'autres théories, il reste à déterminer de tels algorithmes, en particulier dans les théories à classes finies. Les théories algébriques classiques (monoïdes, groupes, anneaux, etc) sont à explorer, aux niveaux théoriques et algorithmiques, avec comme application potentielle la manipulation d'expressions algébriques en Calcul Formel. De même dans le cas logique (anneaux booléens), malgré le nombre a priori infini de généralisations principales, il serait intéressant de pouvoir déterminer rapidement des généralisations aussi maximales que possibles, pour en particulier produire des méthodes d'induction construtive rapide.

F. ANNEXE.

BIBLIOGRAPHIE.

[A.Boudet + E.Contejean 88] "AC Unification is easy", rapport LRI, Univ. Orsay, 1988.

[J.Dixmier 88] "Proof of a conjecture by Erdös and Graham concerning the problem of Frobenius" prépublication, IHES, Bures/Yvette.

[G.Huet 76] "Résolution d'équations dans les langages d'ordre 1...Ω", thèse d'état 1976.

[G.Huet 80] "Confluent reductions: abstract properties and applications to term rewriting systems", J. ACM Vol 27, N° 4, Oct 80, pp 787-821.

[J.M.Hullot 80] "Compilation de formes canoniques dans des théories équationnelles" Thèse de 3° cycle, Université de Paris Sud, centre d'Orsay.

[J.P.Jouannaud 88] communication personnelle.

[P.Le Chenadec 83] "Formes canoniques dans les algèbres finiement présentées", Thèse de 3° cycle, Orsay, 1983.

[G.Plotkin 70] "A note on inductive generalization", Machine Intelligence 5, pp 153-163, Edinburgh University Press ed. 1970.

[L.Pottier 89] "Algorithmes de complétion et généralisation en logique du premier ordre" Thèse de Doctorat Science, Université de Nice, 14 fév. 1989.

[J.C.Reynolds 70] "Transformational systems and the algebraic structure of atomic formulas", Machine Intelligence 5, pp 135-152, Edinburgh University Press ed. 1970.

TABLE DES MATIERES

Imprimé en France
par
l'Institut National de Recherche en Informatique et en Automatique